Blockchain as cryptanalytic tool

Extended Abstract

Manfred Lochter[*]

Bundesamt für Sicherheit in der Informationstechnik (BSI), Germany

Abstract One approach for blockchain based applications to provide a proof-of-work is the computation of hash-values. In our opinion[1] these computations are a waste of energy. It would be highly desirable to find an alternative method that generates useful output. We show how to substitute hashing by performing multiplications on Elliptic Curves in order to find distinguished points that can then be used to solve the discrete logarithm problem on a chosen curve. Today's digital infrastructures rely on only a few curves. We argue that the advent of blockchain based technologies makes the use of only few standardised curves questionable.
In principle all cryptanalytic algorithms that use Rabin's idea of distinguished points can be used in blockchain based attacks. Similar ideas can be used for the number field sieve.

Keywords. Blockchain, Discrete Logarithm, Distinguished Point, Proof-of-Work

1 Introduction

Blockchain based technologies are becoming more and more popular. We do not give a full description of how these technologies work. A good description is e.g. given in [7]. For our purposes it is sufficient to consider a simplified model. We treat blockchains as a linked lists of blocks B_i with the following structure:

$$B_i = \begin{array}{|c|}\hline \text{Pointer to } B_{i-1} \\ \hline \textsc{Data} \\ \hline \text{Proof-of-Work} \\ \hline \end{array}$$

Data can consist of (Bitcoin) transactions, contracts or other data. The most popular instantiation of a blockchain, the cryptocurrency Bitcoin, uses Hash computations to provide a proof-of-work. Let H be a cryptographic hashfunction with n-bit output. Someone who performs work in order to add a block to the blockchain is called a miner. To add a block the miner has to find a seed s such that[2]

$$H(Pointer||Data||s) < S$$

where S is a predefined constant number. This operation can be seen as computing *distinguished hash values.* In real-world applications S is dynamically chosen, such that

[*] Manfred.Lochter@bsi.bund.de
[1] See https://en.bitcoin.it/wiki for a different point of view
[2] We ignore String-To-Integer conversions and padding

on average every 10 minutes a new block will be added to the chain. The miner has to perform on average 2^l hash computations, where

$$l = n - \log_2(S).$$

In January 2017 Bitcoin miners performed about 2.500.000 tera hashes per second (see [2]). This is equivalent to about 2^{86} hashes per year for Bitcoin only. It is interesting to compare with the data for January 2016 (between 750.000 and 1.000.000 tera hashes per second). Today (September 2018) the rate is (according to [1]) 56.537.000 tera hashes per second. This implies that there is an available hash capacity of more than 2^{93} hashes per year within the bitcoin network alone. Assuming a continuation of the trend described above one can assume that the available hashrate will at least double annually for some years.

One of the criticisms of bitcoin is that the hash-calculations do not provide meaningful results. In principle the proof-of-work is a waste of electrical energy. There are, however, some approaches to performing useful scientific computations (see e.g. [7], §8.3).

We introduce a new method for proof-of-work relying on computations that can be used for cryptographic attacks. The computing power usually used for hashing by Bitcoin (and other crypto currencies) is tremendous and can, if used for cryptographic attacks, lead to severe reductions of security for some cryptographic schemes. This is especially the case for applications of Elliptic Curves, where only few standard curves are being used world wide. Considering the price of energy and hardware bitcoin miners spend around 4 billion USD per year on mining [1]. Even nations would not usually spend that amount of money for cryptographic attacks. With blockchain these attacks could come as a windfall profit.

We stress the main point of proof-of-work based schemes is that the work done by the miner must be easy to check. Also cheating miners have to be taken into account.

This paper is organised as follows: We first show how to design a proof-of-work scheme that can be used to attack elliptic curve cryptography. We then briefly describe other applications using collision search. We finish by a modification that can be used during the relation-collection step of the number field sieve.

This paper was first presented at the workshop "The International View of Cryptography and Security and Their Use in Practise", Hong Kong 2017.

2 Elliptic Curves

Let

$$E : Y^2 = X^3 + AX + B$$

be an elliptic curve over a finite prime field $\mathbb{F}_p$ with $p > 3$. By the Hasse-Weil theorem E has $p + 1 + x$ points defined over $GF(p)$, where $|x| \leq 2\sqrt{p}$. For cryptographically strong curves $\#E(\mathbb{F}_p)$ has the form $\#E(\mathbb{F}_p) = \lambda q$, for a prime q and $\lambda \leq 4$. We fix a base point $P \in E(\mathbb{F}_p)$ of order q. Each point Q of order q has the form $Q = mP$ with $m \in \{1, \ldots, q-1\}$.The discrete logarithm problem (DLP) is to find m. Today's digital infrastructures rely heavily on the hardness of the DLP on a small set of elliptic curves. For example Diffie-Hellman key-agreement and signature schemes like ECDSA use elliptic curves[3]. Protocols like TLS limit the number of accepted curves to only a few curves. Also, the security of bitcoin relies on the security of one 256-bit curve.

There are several algorithms to solve instances of the DLP. The most celebrated one being Pollard's Rho algorithm [3] and its variants. This algorithm can be parallelized [8] with linear speed-up using so-called distinguished points. Distinguished points are points with an arithmetic property that can easily be identified. One choice for distinguished points are points whose x-coordinate (interpreted as integer) is smaller than a certain bound, e.g. the most significant DP bits of the x-coordinate equal zero [5]. Each participant then chooses numbers a_0, b_0 and starts iterating on (a_i, b_i) until a distinguished point has been found. There is a vast amount of literature on the optimal choice of iterating functions for the Pollard-Rho-Algorithm.

A standard method is as follows. First divide the set $E(\mathbb{F}_p)$ into three subsets S_1, S_2 and S_3 of almost equal size. Set $G_0 := a_0P + b_0Q$. Then set

$$G_{i+1} := \begin{cases} G_i + P \text{ if } G_i \in S_1 \\ 2G_i \qquad \text{if } G_i \in S_2 \\ G_i + Q \text{ if } G_i \in S_3 \end{cases}$$

Also keep track of the multipliers:

$$(a_{i+1}, b_{i+1}) := \begin{cases} (a_i + 1, b_i) \text{ if } G_i \in S_1 \\ (2a_i, 2b_1) \quad \text{if } G_i \in S_2 \\ (a_i, b_i + 1) \text{ if } G_i \in S_3 \end{cases}$$

The distinguished points are then send to a central server which is used to find collisions $a_iP + b_iQ = a_jP + b_jQ$ from which the discrete logarithm is easily derived. For that purpose the distinguished points are stored[4] in a hash table at an address that is derived from the point. If two distinguished points have to be stored at the same address either a collision has been found, or the data in the corresponding memory location are overwritten. In [8] it is described how to choose optimal parameters (e.g. number of processors, memory size) depending on the price of memory and computation cost. The authors also give an example of how to choose parameters in order to attack the DLP on a 155-bit curve with an budget of 10.000.000$.[5][6] These numbers are from 1996. Compare with the 4 billion USD per year that are spent on mining today.

This approach can be adapted to blockchain based applications. The main obstacle here is that it should be easy to verify that a miner has done work that started at the block B_{i-1}. We propose the following algorithm. Choose a hashfunction whose output

[3] The methods described here can also be applied to systems working in finite fields

[4] To save memory the points are stored in compressed form. I.e. only the non-zero part of the x-coordinate and one bit of the y-coordinate are stored.

[5] They estimate the price of 1MB of memory as $25 (1996). Today the price is about $4 per Gigabyte, and even much lower for harddrives.

[6] 32 days with 333.000 processing units and 16 GB of memory.

fits the bitlength of the curve that shall be attacked. Let RND be a random number[7] To add a new block a miner shall:

1	Initialize a Counter $c = 0$
2	Set $a_0 := H(PointerToB_{i-1}\|\|Data\|\|RND\|\|c)$; $c++$
3	Set $b_0 := H(PointerToB_{i-1}\|\|Data\|\|RND\|\|c)$
4	Iterate on (a_0, b_0) until a distinguished point D has been found, but at most T times
5	If a distinguished point has been found, return PROOF OF WORK $:= (RND, c, D)$,
6	Else $c++$, Update RND; Goto 2.

We can assume that a miner who has found[8] a distinguished point for a counter value c will add it to the chain. The verifier will at most have to perform T steps of the random walk. Thus one can use T and the bound DP defined above to balance the mining process. Typically one would choose a moderately sized T in order to make verification efficient.

In [8] longer chains are considered. This choice of a small T does not influence the performance of the parallel Pollard-Rho algorithm. The reason is that we can see the algorithm above as an alternative definition of a random walk on $E(\mathbb{F}_p)$. The difference to the traditional random walk is that in (a_{kT}, b_{kT}) are defined using a hash function to provide randomness. In this case (a_{kT}, b_{kT}) is independent from (a_{kT-1}, b_{kT-1}). However miners could reserve parts of the RND field for (a_{kT-1}, b_{kT-1}) and update RND accordingly.

Here we do not go into detail about nonces. In principle Bitcoin only supports 32-bit nonces. However the coinbase transaction field can be used as additional nonce. See [7] for more information.

We envision that the RND field is used e.g. to differentiate between different processors, and to be part of the counter. c is the standard nonce.

2.1 Incentives

In typical applications of blockchain based technologies a miner is rewarded for adding a new block. There is no need to reward a user who added a block to an orphaned fork of the chain. Here, however, the computations done by unsuccessful miners are valuable[9] and should be rewarded. There are some possible cases, e. g.:

1. A miner has added a block to the blockchain, but his block is not in the surviving chain.

[7] The reason for introducing below will be given below.
[8] Depending on the setup a miner can be required to contribute more than one distinguished point. In any case (a_0, b_0) should be updated after finding a distinguished point, see [8].
[9] One goal of Bitcoin ist that no miner should have 50 percent ore more of the overall mining capacity

2. A miner has performed computation swithout finding a distinguished point, when a new block B_i is added to the chain by another miner. But he knows that with high probability he will find a distinguished point soon[10]. In this case he should be able to carry on computing for a certain amount of time and to add a block B'_i after B_{i-1}. This block is known to become orphaned. But the miner should be able to claim a reward that is proportional to the reward for adding B_i.
3. Alternatively, unsuccessful miners could just sell their distinguished points to the attacker.

There are papers that discuss incentives for creating orphaned blocks in more detail, e.g. [9]

2.2 Mining pools and multiprocessor machines

Up to now we have described a simplified model that assumes that individual miners are searching for distinguished points. In reality groups of miners will work together and use multiple processors. In order to distribute work between members of a pool and between processors one could replace the simple counter, described above by a combination of a random number RND and a counter. The generation of RND would be left to the miners.

2.3 Secrecy and multiple discrete logarithms

An attacker using the method described above need not reveal which discrete logarithm he is searching for. He can just blind the point Q by providing a known multiple τQ as challenge.

The method described can be extended to multiple discrete logarithms, see [6]. Note that as a by-product trust in any elliptic curve could breeak down if it becomes part of a proof-of-work scheme as scetched here.

Interestingly this process would not apply to Bitcoin itself. Bitcoin adresses are derived from a private key via double hashing with SHA-256 and RIPEMD-160. This means that roughly 2^{96} private keys lead to the same bitcoin address. An attacker would only have to recover one of these possible keys. This could be done by first finding a curve point that hashes to the address and then to find the discrete logarithm of this point.

The owner of a bitcoin has to sign transactions with a secret key that fits the address of the BTC that is transferred. This allows for a direct attack with Pollards Rho algorithm or with Shors algorithm on a quantum computer. In this case the attacker would have only a small time-window for a double-spend attack.

[10] We assume that a miner uses many processors in parallel and thus the time expected for finding a distinguished point has very small variance. A "miner" in this sense could e.g. be a mining pool. For simplicity we also assume that only one distinguished point has to be contributed by the miner for adding a block

2.4 Collision attacks and meet-in-the-middle attacks

[8] gives many other cryptographic applications of of parallel collision search, but see also [4].These methods can also be combined with blockchains if the system setup allows to choose a common attack goal and to identify and sanction cheaters.

3 Number field sieve

Similar techniques can be applied to turn blockchain into a tool to find relations that can be used with the number field sieve. The main idea of the number field sieve is to sieve through an area of numbers and find relations, i.e. to find many pairs (a, b) of integers such that the value $f(a + b\alpha)$ is PB_1 smooth, up to two or three larger prime factors that are smaller than a bound PB_2. Obviously it is easier to check a relation than to find a relation. However, a miner could try to cheat by just working on function values that can easily be factored. He would thus only contribute relations where $f(a + b\alpha)$ is a product of small primes. From previous experiments with the NFS one already knows the distribution of relations that can be expected from an honest miner. Therefore in the NFS case the proof-of-work would consist of two stages.

- Check relations.
- Check quality of relations.

As before the search area of the miner would be determined by the input data.

4 Conclusion

We have shown that the computational power used for cryptocurrencies could also be used for parallel cryptographic attacks, especially against systems where only few parameter sets are used.

References

1. `https://digiconomist.net/bitcoin-energy-consumption`. [Online; accessed 26-August 2018].
2. `https://blockchain.info/charts/hash-rate`, 2017. [Online; accessed 3-January-2017].
3. S. D. Galbraith. *Mathematics of public key cryptography.*
4. J. Hong, G. W. Lee, and D. Ma. Analysis of the parallel distinguished point tradeoff. Cryptology ePrint Archive, Report 2011/387, 2011. `https://eprint.iacr.org/2011/387`.
5. A. Joux. *Algorithmic Cryptanalysis.* Chapman & Hall/CRC, 1st edition, 2009.
6. F. Kuhn and R. Struik. Random walks revisited: Extensions of pollard's rho algorithm for computing multiple discrete logarithms. In S. Vaudenay and A. M. Youssef, editors, *Selected Areas in Cryptography*, pages 212–229, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
7. A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, Princeton, NJ, USA, 2016.
8. P. C. V. Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12:1–28, 1999.
9. Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. Cryptology ePrint Archive, Report 2013/881, 2013. `https://eprint.iacr.org/2013/881`.